Evaluating Personal Job Market Prospects in 2024

Final Report

Furong Wang

Marco Perez

April 30, 2025

# Home $

title: “About”

This site aims to cover a profound investigation related to Trends in Salaries and Compensations in AI vs non AI careers

# Introduction

title: “Salary & Compensation Trends in AI vs. Non-AI Careers” author: “Furong Wang, Marco Perez Garcia” date: today format: html: bibliography: references.bib csl: csl/econometrica.csl toc: true

Recent research has highlighted a growing divergence in salary trends between artificial intelligence (AI)-focused careers and more traditional data science roles. Zhu ((2024)) found that professionals specializing in AI-related fields, such as machine learning engineers and AI researchers, consistently command higher salaries than their non-AI counterparts, including data analysts and general data scientists. This difference in compensation reflects the increasing demand for AI expertise as industries integrate automation, deep learning, and predictive analytics into their operations. While AI roles require specialized skills in areas such as neural networks and natural language processing, traditional data science positions often focus more on business intelligence, statistical analysis, and data visualization, which though even that is valuable it does not see the same salary premiums.

Other studies reinforce this trend, showing how company size and industry specialization further impact salary structures. Chen, Song, and Lam ((2024)) analyzed U.S. salary trends from 2020 to 2023, reporting that salaries in AI-driven roles have shown steady increases, particularly within mid-to-large tech companies investing in AI innovation. In contrast, non-AI data science roles, such as data analysts, have experienced slower growth, and some projections indicate potential stagnation or slight salary declines in 2024. Similarly, Quan and Raheem ((2023)) found that professionals with expertise in AI, cloud computing, and big data technologies earn higher salaries than those with more generalist skills. Their findings suggest that as AI adoption expands across industries, the wage gap between AI and non-AI roles may continue to grow, emphasizing the importance of specialized technical expertise for long-term career advancement in data science.

## References

# Data Cleaning

title: Data Cleaning & Exploration subtitle: Comprehensive Data Cleaning & Exploratory Analysis of Job Market Trends author: - name: Furong Wang affiliations: - id: bu name: Boston University city: Boston state: MA - name: Marco Perez Garcia affiliations: - ref: bu bibliography: references.bib csl: csl/econometrica.csl format: html: toc: true number-sections: true df-print: paged jupyter: python3

# **Data Cleaning & Preprocessing**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import missingno as msno  
import plotly.express as px

## Dropping Unnecessary Columns

### To simplify our job market analysis, we need to drop columns that are either unnecessary, duplicated, or outdated.

Specifically, we removed:  
- **Older NAICS and SOC codes (e.g., NAICS2, SOC\_2).**  
The North American Industry Classification System (NAICS) and Standard Occupational Classification (SOC) systems undergo periodic updates. Retaining only NAICS\_2022\_6 and SOC\_2021\_4 ensures we use the most recent classification standards. Moreover, older codes are redundant and may lead to inconsistencies in trend analysis.  
- **Tracking data and URLs (e.g., ID, DUPLICATES).**  
These columns related to data collection timestamps, unique identifiers, or internal system references, which do not contribute to meaningful insights about the job market. Similarly, URLs are not necessary for our analysis as they do not provide any additional value or context but add unnecessary complexity to the dataset.

### Why removing multiple versions of NAICS/SOC codes?

There are some reasons for this. Firstly, keeping only the latest industry and occupation classifications ensures our analysis reflects the most recent classification standards and avoid confusion and inconsistencies in classification. Additionally, reducing unnecessary columns speeds up data processing and enhances readability. This is particularly important when working with large datasets, as it minimizes the risk of errors and improves the efficiency of our analysis. Finally, it helps to focus on the most relevant information, allowing for clearer insights and conclusions regarding job market trends.

### Impact on Analysis:

By removing outdated and irrelevant columns, we achieve:  
- More accurate job market trends, focusing on meaningful variables.  
- Easier interpretation without clutter from redundant or technical fields.  
- Faster analysis and visualization, improving overall efficiency.

job\_postings = pd.read\_csv('lightcast\_job\_postings.csv')

/tmp/ipykernel\_2773/3483734598.py:1: DtypeWarning:  
  
Columns (19,30) have mixed types. Specify dtype option on import or set low\_memory=False.

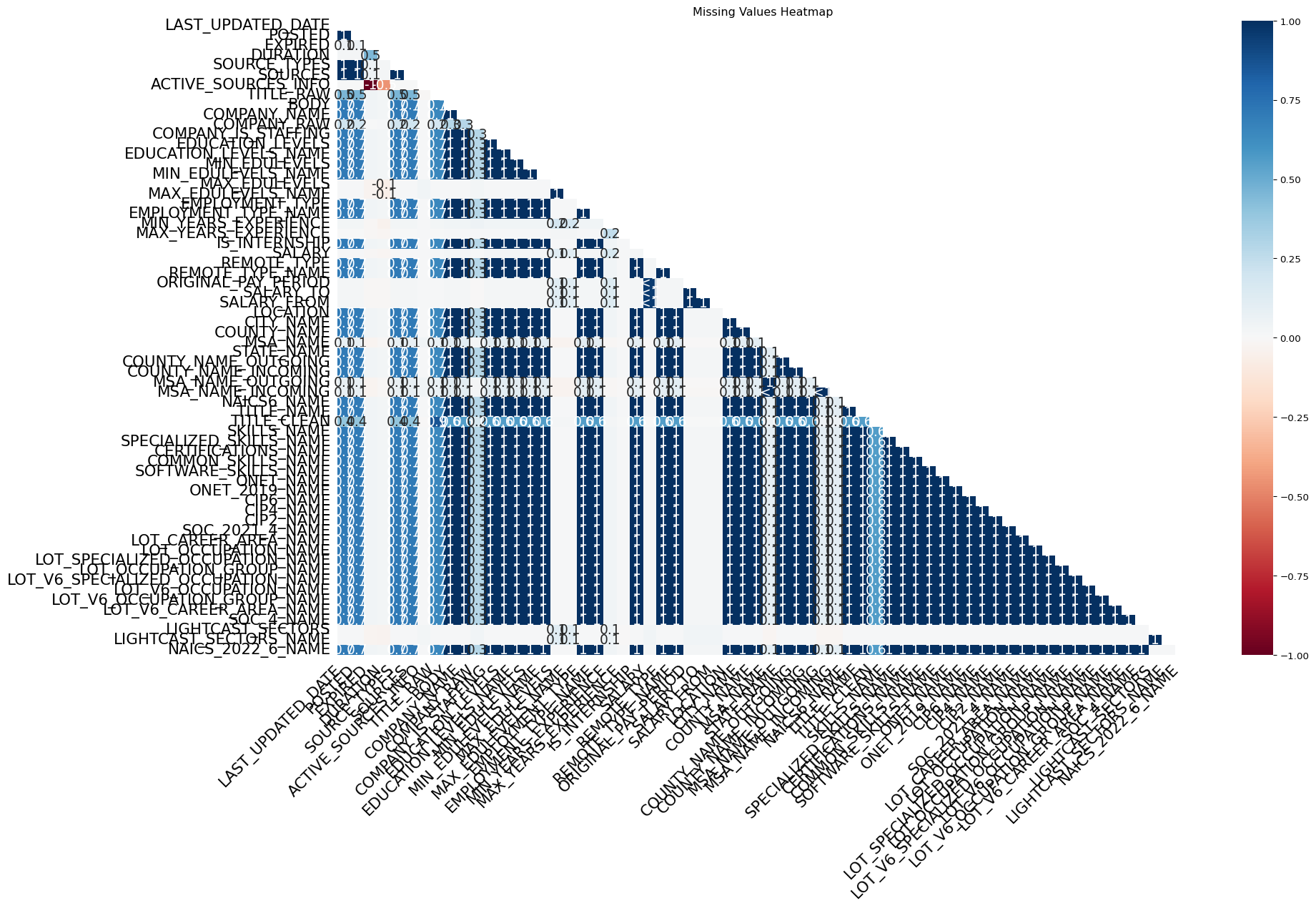
columns\_to\_drop = [  
 "ID", "URL", "ACTIVE\_URLS", "DUPLICATES", "LAST\_UPDATED\_TIMESTAMP", "ACTIVE\_URLS", "TITLE", "COMPANY",  
 "MSA", "STATE", "COUNTY", "CITY", "COUNTY\_OUTGOING", "COUNTY\_INCOMING", "MSA\_OUTGOING", "MSA\_INCOMING",  
 "ONET", "ONET\_2019", "CIP2", "CIP4", "CIP6", "MODELED\_DURATION", "MODELED\_EXPIRED",  
 "CERTIFICATIONS", "COMMON\_SKILLS", "SPECIALIZED\_SKILLS", "SKILLS", "SOFTWARE\_SKILLS",  
 "LOT\_V6\_CAREER\_AREA", "LOT\_V6\_OCCUPATION\_GROUP", "LOT\_V6\_OCCUPATION", "LOT\_V6\_SPECIALIZED\_OCCUPATION",  
 "LOT\_OCCUPATION\_GROUP", "LOT\_SPECIALIZED\_OCCUPATION", "LOT\_OCCUPATION", "LOT\_CAREER\_AREA",  
 "NAICS2", "NAICS2\_NAME", "NAICS3", "NAICS3\_NAME", "NAICS4", "NAICS4\_NAME", "NAICS5", "NAICS5\_NAME", "NAICS6",  
 "NAICS\_2022\_2", "NAICS\_2022\_2\_NAME", "NAICS\_2022\_3", "NAICS\_2022\_3\_NAME", "NAICS\_2022\_4", "NAICS\_2022\_4\_NAME",  
 "NAICS\_2022\_5", "NAICS\_2022\_5\_NAME", "NAICS\_2022\_6",  
 "SOC\_2", "SOC\_2\_NAME", "SOC\_3", "SOC\_3\_NAME", "SOC\_5", "SOC\_5\_NAME", "SOC\_4",  
 "SOC\_2021\_2", "SOC\_2021\_2\_NAME", "SOC\_2021\_3", "SOC\_2021\_3\_NAME", "SOC\_2021\_5", "SOC\_2021\_5\_NAME", "SOC\_2021\_4"  
]  
  
job\_postings.drop(columns = columns\_to\_drop, inplace = True)

## Handling Missing Values

### We used different strategies for missing values:

plt.figure(figsize=(8, 6))  
msno.heatmap(job\_postings)  
plt.title("Missing Values Heatmap")  
plt.show()

<Figure size 768x576 with 0 Axes>



#### Dealing with the Salary Column

The SALARY column has a significant number of missing values. To handle this, we replaced the missing values with the median salary for that specific title or industry. This approach is effective because it minimizes the impact of outliers and provides a more accurate representation of the typical salary for each job title.

title\_median\_salary = job\_postings.groupby('TITLE\_NAME')['SALARY'].median()  
industry\_median\_salary = job\_postings.groupby('NAICS\_2022\_6\_NAME')['SALARY'].median()

job\_postings['SALARY'] = job\_postings.apply(  
 lambda row: title\_median\_salary[row['TITLE\_NAME']]  
 if pd.isna(row['SALARY']) and row['TITLE\_NAME'] in title\_median\_salary else row['SALARY'],   
 axis=1  
)

job\_postings['SALARY'] = job\_postings.apply(  
 lambda row: industry\_median\_salary[row['NAICS\_2022\_6\_NAME']]  
 if pd.isna(row['SALARY']) and row['NAICS\_2022\_6\_NAME'] in industry\_median\_salary else row['SALARY'],   
 axis=1  
)

job\_postings['SALARY'] = job\_postings['SALARY'].fillna(job\_postings["SALARY"].median())

#### Dealing with Columns with >50% missing values

Dealing with columns that have more than 50% missing values is crucial for maintaining the integrity of our dataset. Columns with excessive missing data can introduce bias and reduce the reliability of our analysis. Therefore, we removed any columns that exceed this threshold. This ensures that our dataset remains focused on relevant and reliable information, enhancing the quality of our insights.

job\_postings.dropna(thresh = len(job\_postings) \* 0.5, axis = 1, inplace = True)

#### Dealing with Categorical fields

Categorical fields, such as TITLE\_RAW, were filled with “Unknown” for missing values. This approach allows us to retain the integrity of the dataset without introducing bias from arbitrary values. By labeling missing categorical data as “Unknown”, we can still analyze trends without losing valuable information.

job\_postings['TITLE\_RAW'] = job\_postings['TITLE\_RAW'].fillna("Unknown")  
job\_postings['TITLE\_CLEAN'] = job\_postings['TITLE\_CLEAN'].fillna("Unknown")  
job\_postings['COMPANY\_RAW'] = job\_postings['COMPANY\_RAW'].fillna("Unknown")  
job\_postings['MSA\_NAME'] = job\_postings['MSA\_NAME'].fillna("Unknown")  
job\_postings['MSA\_NAME\_OUTGOING'] = job\_postings['MSA\_NAME\_OUTGOING'].fillna("Unknown")  
job\_postings['MSA\_NAME\_INCOMING'] = job\_postings['MSA\_NAME\_INCOMING'].fillna("Unknown")

#### Dealing with Datetime fields

For the EXPIRED variable, we chose to fill the missing values with the maximum date from this column. We assumed that the missing value here is because the post has not expired yet. By using the maximum date, we can effectively handle missing values without introducing bias or skewing the results.

job\_postings['POSTED'] = pd.to\_datetime(job\_postings['POSTED'])  
job\_postings['EXPIRED'] = pd.to\_datetime(job\_postings['EXPIRED'])

max\_expired\_date = job\_postings['EXPIRED'].max()  
job\_postings['EXPIRED'] = job\_postings['EXPIRED'].fillna(max\_expired\_date)

#### Dealing with Numerical fields

For the MIN\_YEARS\_EXPERIENCE variable, we chose to fill the missing values with the median MIN\_YEARS\_EXPERIENCE for a specific title or industry, similar to how we did with the SALARY variable. This can minimize the impact of outliers and provides a more accurate representation of the typical years of experience required for each job title.

title\_median\_exp = job\_postings.groupby('TITLE\_NAME')['MIN\_YEARS\_EXPERIENCE'].median()  
industry\_median\_exp = job\_postings.groupby('NAICS\_2022\_6\_NAME')['MIN\_YEARS\_EXPERIENCE'].median()

job\_postings['MIN\_YEARS\_EXPERIENCE'] = job\_postings.apply(  
 lambda row: title\_median\_exp[row['TITLE\_NAME']]  
 if pd.isna(row['MIN\_YEARS\_EXPERIENCE']) and row['TITLE\_NAME'] in title\_median\_exp else row['MIN\_YEARS\_EXPERIENCE'],   
 axis=1  
)

job\_postings['MIN\_YEARS\_EXPERIENCE'] = job\_postings.apply(  
 lambda row: industry\_median\_exp[row['NAICS\_2022\_6\_NAME']]  
 if pd.isna(row['MIN\_YEARS\_EXPERIENCE']) and row['NAICS\_2022\_6\_NAME'] in industry\_median\_exp else row['MIN\_YEARS\_EXPERIENCE'],   
 axis=1  
)

job\_postings['MIN\_YEARS\_EXPERIENCE'] = job\_postings['MIN\_YEARS\_EXPERIENCE'].fillna(job\_postings["MIN\_YEARS\_EXPERIENCE"].median())

DURATION variable is also a numerical field, but it has a different approach. We will fill the missing values with the difference between the POSTED and EXPIRED, which calculates the actual time span based on the available dates.

def impute\_duration(cols):  
 posted = cols.iloc[0]  
 expired = cols.iloc[1]  
 duration = cols.iloc[2]  
  
 if pd.isnull(duration):  
 return expired - posted  
 else:   
 return duration

job\_postings['DURATION'] = job\_postings[['POSTED', 'EXPIRED', 'DURATION']].apply(impute\_duration, axis = 1)

## Removing Duplicate Job Postings

### To ensure each job is counted only once, we removed duplicates based on job title, company, location, and posting date.

job\_postings = job\_postings.drop\_duplicates(subset=["TITLE\_NAME", "COMPANY\_NAME", "LOCATION", "POSTED"], keep = "first")

job\_postings.to\_csv('job\_postings\_cleaned.csv', index=False)

# **Exploratory Data Analysis (EDA)**

## Top 20 companies by job postings

filtered\_companies = job\_postings[job\_postings["COMPANY\_NAME"] != "Unclassified"]  
  
top\_companies = filtered\_companies["COMPANY\_NAME"].value\_counts().head(20)  
  
fig = px.bar(  
 x=top\_companies.values,  
 y=top\_companies.index,  
 orientation='h',  
 title="Top 20 Companies by Job Postings (Excluding Unclassified)",  
 labels={'x': 'Number of Job Postings', 'y': 'Company Name'},  
 text=top\_companies.values  
)  
  
fig.update\_layout(  
 xaxis\_title="Number of Job Postings",  
 yaxis\_title="Company",  
 yaxis={'categoryorder': 'total ascending'},   
 height=600,   
 width=900  
)  
  
fig.show()

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): text/html

The visualization of the top 20 companies by job postings (excluding “Unclassified”) highlights key trends in the job market, particularly in the increasing demand for AI-related roles. Many of the companies with the most postings—Deloitte, Accenture, PricewaterhouseCoopers (PwC), Oracle, Infosys, Meta, and CDW—are major players in technology, consulting, and digital transformation, sectors that have been heavily investing in AI, machine learning, and data-driven innovation.

The dominance of these companies in job postings suggests that careers in AI and technology-related fields are in high demand. Consulting giants like Deloitte, Accenture, PwC, and KPMG are actively expanding their AI divisions, helping businesses integrate AI into their operations. For instance, Deloitte has launched several AI tools, including chatbots like “DARTbot” for audit professionals and “NavigAite” for document review, to enhance efficiency and client services (Stokes, 2025). Additionally, companies like Meta are pioneers in AI research, focusing on areas such as generative AI, automation, and data science. Even in non-tech sectors, financial and healthcare firms such as Citigroup, Cardinal Health, and Blue Cross Blue Shield are leveraging AI for fraud detection, risk assessment, and personalized healthcare.

These trends indicate that pursuing a career in AI-related fields, such as data science, machine learning engineering, and AI research, could provide greater job opportunities and higher earning potential. The strong presence of technology and consulting firms in job postings reflects how AI is becoming a fundamental part of business strategies across industries. While traditional, non-AI careers will continue to exist, the rapid push toward automation and intelligent systems suggests that AI-related skills will be increasingly valuable in both technical and non-technical roles. As industries continue adopting AI, professionals who develop expertise in this area may have a competitive advantage in the evolving job market.

## Salary Distribution by Industry

fig = px.box(job\_postings, x="NAICS\_2022\_6\_NAME", y="SALARY", title="Salary Distribution by Industry")  
fig.update\_layout(width=1200, height=1000)  
fig.show()

Unable to display output for mime type(s): text/html

The box plot provides a clearer view of salary distributions across industries, highlighting variations in median salaries and outliers. Most industries exhibit salary concentrations below $200K, with some sectors showing significantly higher outliers above $300K-$500K, suggesting high-paying roles in specialized fields.

AI-related jobs, typically found in industries such as technology, finance, and advanced manufacturing, often contribute to these high-salary outliers. Roles in machine learning, data science, and artificial intelligence engineering command premium salaries due to their specialized skill requirements, talent scarcity, and high demand across multiple industries. The broader salary spread in AI-intensive fields may also reflect differences in job seniority, from entry-level analysts to highly compensated AI researchers and executives.

Additionally, AI-driven industries tend to offer competitive compensation to attract top talent, given the rapid pace of technological advancement and the strategic importance of AI in business growth. The dense clustering of lower salaries in non-AI industries indicates a more constrained range, potentially due to standardized pay structures or lower technical barriers to entry.

## Top 5 Occupations by Average Salary

avg\_salary\_per\_occupation = job\_postings.groupby("LOT\_V6\_OCCUPATION\_NAME")["SALARY"].mean().reset\_index()  
  
top\_occupations = avg\_salary\_per\_occupation.sort\_values(by="SALARY", ascending=False).head(5)  
  
fig = px.bar(  
 top\_occupations,  
 x="SALARY",  
 y="LOT\_V6\_OCCUPATION\_NAME",  
 orientation='h',  
 title="Top 5 Occupations by Average Salary",  
 labels={"SALARY": "Average Salary ($)", "LOT\_V6\_OCCUPATION\_NAME": "Occupation"},  
 text=top\_occupations["SALARY"]  
 )  
  
fig.update\_layout(  
 xaxis\_title="Average Salary ($)",  
 yaxis\_title="Occupation",  
 yaxis={"categoryorder": "total ascending"},   
 height=700,  
 width=900  
 )  
  
fig.show()

Unable to display output for mime type(s): text/html

The salary distribution in the graph clearly shows that the highest-paying occupations are directly tied to artificial intelligence, data analytics, and business intelligence. The top-paying role, “Computer Systems Engineer / Architect,” averages over $156,000, followed by “Business Intelligence Analyst” at $125,000 and other AI-driven roles like “Data Mining Analyst” and “Market Research Analyst,” all exceeding $100,000. These occupations rely heavily on AI, machine learning, and data-driven decision-making, making it clear that mastering AI-related skills is directly linked to higher salaries. The strong earnings for these roles indicate that industries are willing to pay a premium for professionals who can build, interpret, and optimize AI-driven systems.

In contrast, traditional non-AI careers, which are not as data or automation-focused, tend to fall outside these top salary brackets. The job market is shifting towards AI dependency, where knowing how to work with artificial intelligence, big data, and automation tools is no longer just an advantage but a necessity for higher-paying opportunities. As industries integrate AI at an increasing pace, professionals who fail to develop AI-related expertise risk stagnating in lower-paying roles, while those who embrace AI technologies position themselves for significantly better financial rewards.

# Exploratory Data Analysis (EDA)

title: Exploratory Data Analysis subtitle: Enhance EDA with Improved Visualizations and Deeper Insights author: - name: Furong Wang affiliations: - id: bu name: Boston University city: Boston state: MA - name: Marco Perez Garcia affiliations: - ref: bu bibliography: references.bib csl: csl/econometrica.csl format: html: toc: true number-sections: true df-print: paged jupyter: python3

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import missingno as msno  
import plotly.express as px

job\_postings = pd.read\_csv('job\_postings\_cleaned.csv')

# **Exploratory Data Analysis & Visualization**

plotly\_layout = dict(  
 font=dict(family="Arial", size=14),  
 title\_font=dict(size=20, family="Arial", color="black"),  
 paper\_bgcolor="white",  
 plot\_bgcolor="white",  
 margin=dict(t=60, l=60, r=30, b=60),  
 legend=dict(bordercolor="lightgray", borderwidth=1),  
 xaxis=dict(  
 title\_font=dict(size=16, color="black"),   
 tickfont=dict(size=12, color="black"),   
 showgrid=True,   
 gridcolor="lightgray",  
 showline=True,  
 linecolor="black",  
 linewidth=1,  
 mirror=True,  
 zeroline=True,  
 zerolinecolor="gray",  
 zerolinewidth=1  
 ),  
 yaxis=dict(  
 title\_font=dict(size=16, color="black"),   
 tickfont=dict(size=12, color="black"),   
 showgrid=True,   
 gridcolor="lightgray",  
 showline=True,  
 linecolor="black",  
 linewidth=1,  
 mirror=True,  
 zeroline=True,  
 zerolinecolor="gray",  
 zerolinewidth=1  
 ),  
)

## Top 20 companies by job postings

filtered\_companies = job\_postings[job\_postings["COMPANY\_NAME"] != "Unclassified"]  
  
top\_companies = filtered\_companies["COMPANY\_NAME"].value\_counts().head(20)  
  
fig = px.bar(  
 x=top\_companies.values,  
 y=top\_companies.index,  
 orientation='h',  
 title="Top 20 Companies by Job Postings (Excluding Unclassified)",  
 labels={'x': 'Number of Job Postings', 'y': 'Company Name'},  
 text=top\_companies.values  
)  
  
fig.update\_layout(  
 xaxis\_title="Number of Job Postings",  
 yaxis\_title="Company",  
 yaxis={'categoryorder': 'total ascending'},   
 height=600,   
 width=900  
)  
  
fig.update\_layout(\*\*plotly\_layout)   
fig.show()

Unable to display output for mime type(s): text/html

The visualization of the top 20 companies by job postings (excluding “Unclassified”) highlights key trends in the job market, particularly in the increasing demand for AI-related roles. Many of the companies with the most postings—Deloitte, Accenture, PricewaterhouseCoopers (PwC), Oracle, Infosys, Meta, and CDW—are major players in technology, consulting, and digital transformation, sectors that have been heavily investing in AI, machine learning, and data-driven innovation.

The dominance of these companies in job postings suggests that careers in AI and technology-related fields are in high demand. Consulting giants like Deloitte, Accenture, PwC, and KPMG are actively expanding their AI divisions, helping businesses integrate AI into their operations. For instance, Deloitte has launched several AI tools, including chatbots like “DARTbot” for audit professionals and “NavigAite” for document review, to enhance efficiency and client services (Stokes, 2025). Additionally, companies like Meta are pioneers in AI research, focusing on areas such as generative AI, automation, and data science. Even in non-tech sectors, financial and healthcare firms such as Citigroup, Cardinal Health, and Blue Cross Blue Shield are leveraging AI for fraud detection, risk assessment, and personalized healthcare.

These trends indicate that pursuing a career in AI-related fields, such as data science, machine learning engineering, and AI research, could provide greater job opportunities and higher earning potential. The strong presence of technology and consulting firms in job postings reflects how AI is becoming a fundamental part of business strategies across industries. While traditional, non-AI careers will continue to exist, the rapid push toward automation and intelligent systems suggests that AI-related skills will be increasingly valuable in both technical and non-technical roles. As industries continue adopting AI, professionals who develop expertise in this area may have a competitive advantage in the evolving job market.

## Salary Distribution by Industry

fig = px.box(job\_postings, x="NAICS\_2022\_6\_NAME", y="SALARY", title="Salary Distribution by Industry")  
fig.update\_layout(width=1200, height=1000)  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

Unable to display output for mime type(s): text/html

The box plot provides a clearer view of salary distributions across industries, highlighting variations in median salaries and outliers. Most industries exhibit salary concentrations below $200K, with some sectors showing significantly higher outliers above $300K-$500K, suggesting high-paying roles in specialized fields.

AI-related jobs, typically found in industries such as technology, finance, and advanced manufacturing, often contribute to these high-salary outliers. Roles in machine learning, data science, and artificial intelligence engineering command premium salaries due to their specialized skill requirements, talent scarcity, and high demand across multiple industries. The broader salary spread in AI-intensive fields may also reflect differences in job seniority, from entry-level analysts to highly compensated AI researchers and executives.

Additionally, AI-driven industries tend to offer competitive compensation to attract top talent, given the rapid pace of technological advancement and the strategic importance of AI in business growth. The dense clustering of lower salaries in non-AI industries indicates a more constrained range, potentially due to standardized pay structures or lower technical barriers to entry.

## Top 5 Occupations by Average Salary

avg\_salary\_per\_occupation = job\_postings.groupby("LOT\_V6\_OCCUPATION\_NAME")["SALARY"].mean().reset\_index()  
  
top\_occupations = avg\_salary\_per\_occupation.sort\_values(by="SALARY", ascending=False).head(5)  
  
fig = px.bar(  
 top\_occupations,  
 x="SALARY",  
 y="LOT\_V6\_OCCUPATION\_NAME",  
 orientation='h',  
 title="Top 5 Occupations by Average Salary",  
 labels={"SALARY": "Average Salary ($)", "LOT\_V6\_OCCUPATION\_NAME": "Occupation"},  
 text=top\_occupations["SALARY"]  
 )  
  
fig.update\_layout(  
 xaxis\_title="Average Salary ($)",  
 yaxis\_title="Occupation",  
 yaxis={"categoryorder": "total ascending"},   
 height=700,  
 width=900  
 )  
  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

Unable to display output for mime type(s): text/html

The salary distribution in the graph clearly shows that the highest-paying occupations are directly tied to artificial intelligence, data analytics, and business intelligence. The top-paying role, “Computer Systems Engineer / Architect,” averages over $156,000, followed by “Business Intelligence Analyst” at $125,000 and other AI-driven roles like “Data Mining Analyst” and “Market Research Analyst,” all exceeding $100,000. These occupations rely heavily on AI, machine learning, and data-driven decision-making, making it clear that mastering AI-related skills is directly linked to higher salaries. The strong earnings for these roles indicate that industries are willing to pay a premium for professionals who can build, interpret, and optimize AI-driven systems.

In contrast, traditional non-AI careers, which are not as data or automation-focused, tend to fall outside these top salary brackets. The job market is shifting towards AI dependency, where knowing how to work with artificial intelligence, big data, and automation tools is no longer just an advantage but a necessity for higher-paying opportunities. As industries integrate AI at an increasing pace, professionals who fail to develop AI-related expertise risk stagnating in lower-paying roles, while those who embrace AI technologies position themselves for significantly better financial rewards.

## **Enhanced Visualizations**

## Job Postings Trend Over Time (Top Companies)

job\_postings['POSTED'] = pd.to\_datetime(job\_postings['POSTED'])  
top\_companies = (  
 job\_postings[job\_postings["COMPANY\_NAME"] != "Unclassified"]["COMPANY\_NAME"]  
 .value\_counts()  
 .head(10)  
 .index  
)  
  
filtered = job\_postings[job\_postings['COMPANY\_NAME'].isin(top\_companies)]  
  
trend = (  
 filtered.groupby([filtered['POSTED'].dt.to\_period('M'), 'COMPANY\_NAME'])  
 .size()  
 .reset\_index(name='Postings')  
)  
trend['POSTED'] = trend['POSTED'].dt.to\_timestamp()  
  
fig = px.line(trend, x='POSTED', y='Postings', color='COMPANY\_NAME',  
 title='Monthly Job Postings for Top 10 Companies')  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

Unable to display output for mime type(s): text/html

The line chart above reveals dynamic shifts in job posting activity among the top 10 hiring companies over recent months. Several key patterns emerge:

* Infosys shows a strong upward trend, indicating a possible expansion phase or increased demand for tech-related talent. This could reflect growing project loads or client demand in IT services and consulting.
* Accenture and Deloitte maintain relatively stable posting volumes, suggesting consistent hiring pipelines. This stability aligns with their roles as global consulting giants with ongoing needs for specialized talent in digital transformation, data analytics, and strategy.
* Humana and Insight Global exhibit moderate declines followed by slight recoveries, potentially pointing to seasonal or project-based hiring fluctuations in healthcare and staffing services.
* Companies like KPMG, Oracle, and PricewaterhouseCoopers (PwC) show lower and flatter posting trends, possibly indicating a more conservative hiring approach or specific recruitment periods during the year.
* Merit America, a nonprofit focused on career advancement, remains on the lower end of the spectrum. However, its presence in the top 10 indicates consistent demand in educational or workforce development roles.

Overall, the chart highlights Infosys as a standout, with its consistent rise suggesting aggressive recruitment. In contrast, other firms maintain steady or slightly fluctuating volumes, reflecting industry-specific hiring cycles. This trend-based view can be valuable for job seekers, workforce planners, or analysts studying labor market activity in the consulting, healthcare, tech, and staffing sectors.

## Salary Distribution by Industry (Filtered Outliers)

Q1 = job\_postings['SALARY'].quantile(0.25)  
Q3 = job\_postings['SALARY'].quantile(0.75)  
IQR = Q3 - Q1  
  
filtered\_salaries = job\_postings[  
 (job\_postings['SALARY'] >= Q1 - 1.5\*IQR) &   
 (job\_postings['SALARY'] <= Q3 + 1.5\*IQR)  
]  
  
fig = px.box(filtered\_salaries, x="NAICS\_2022\_6\_NAME", y="SALARY",   
 title="Filtered Salary Distribution by Industry")  
fig.update\_layout(width=1200, height=800, xaxis\_tickangle=45)  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

Unable to display output for mime type(s): text/html

The box plot above provides a cleaned and focused view of salary distributions across different industries, with extreme outliers removed to highlight more meaningful central trends.

* High variation across industries: Some industries display a narrow salary band, suggesting standardized roles (e.g., Retail or Administrative sectors), while others—especially in tech, consulting, and finance—show wider spreads, indicating diverse job levels and pay scales.
* Technology and data-driven sectors (e.g., Computer Systems Design, Custom Software Development) tend to cluster toward the higher end of the salary spectrum, reflecting the premium placed on digital skills, AI, and advanced analytics.
* Healthcare and scientific industries also show strong mid-to-upper ranges, hinting at specialized roles that demand advanced education or certifications.
* In contrast, industries like Warehousing, Food Services, and Retail generally reflect lower median salaries, consistent with roles requiring less formal education or technical expertise.

This visualization emphasizes how industry selection can significantly impact earning potential, even before considering role or experience level. For job seekers or workforce planners, it provides a valuable benchmark when evaluating career paths or advising on industry transitions.

## Fastest-Growing Industries Over Time

monthly\_industry = (  
 job\_postings.groupby([job\_postings['POSTED'].dt.to\_period("M"), "NAICS\_2022\_6\_NAME"])  
 .size()  
 .reset\_index(name='Postings')  
)  
monthly\_industry["POSTED"] = monthly\_industry["POSTED"].dt.to\_timestamp()  
  
top\_industries = monthly\_industry.groupby("NAICS\_2022\_6\_NAME")["Postings"].sum().nlargest(6).index  
  
top\_industries = [industry for industry in top\_industries if industry != "Unclassified Industry"]  
  
filtered\_growth = monthly\_industry[monthly\_industry["NAICS\_2022\_6\_NAME"].isin(top\_industries)]  
  
fig = px.line(filtered\_growth, x="POSTED", y="Postings", color="NAICS\_2022\_6\_NAME",  
 title="Top 5 Industries by Job Postings Over Time (Excluding Unclassified)")  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

Unable to display output for mime type(s): text/html

This line plot presents job posting trends across the top five industries (excluding unclassified roles), offering a clearer picture of sector-specific hiring momentum over the past several months.

* Employment Placement Agencies show the most significant increase in job postings, suggesting a surge in demand for staffing services. This could reflect broader labor market activity, such as rising contract work, workforce mobility, or seasonal hiring cycles.
* Administrative Management and Consulting Services maintain consistently high levels of postings, highlighting the ongoing demand for business strategy, operations, and project management talent. The slight upward trend may align with businesses seeking advisory support during periods of uncertainty or transformation.
* Computer Systems Design Services and Custom Computer Programming Services demonstrate steady hiring activity, reinforcing the continued need for tech infrastructure, custom software development, and IT support roles across industries.
* Commercial Banking, while slightly more volatile, remains a key hiring industry. This might reflect fluctuations in financial service needs, regulatory adjustments, or regional economic conditions.

Overall, the chart illustrates that technology, consulting, staffing, and finance remain dominant hiring sectors — with tech-related industries showing stable demand and staffing services accelerating most rapidly. These insights are valuable for job seekers targeting high-opportunity industries, and for workforce planners aiming to align talent strategies with real-time market shifts.

## Salary Trends Over Time for Top 5 Occupations

job\_postings['POSTED'] = pd.to\_datetime(job\_postings['POSTED'])  
top\_occ = job\_postings['LOT\_V6\_OCCUPATION\_NAME'].value\_counts().head(5).index  
  
filtered\_jobs = job\_postings[job\_postings['LOT\_V6\_OCCUPATION\_NAME'].isin(top\_occ)]  
filtered\_jobs['Month'] = filtered\_jobs['POSTED'].dt.to\_period("M").dt.to\_timestamp()  
  
salary\_trend = (  
 filtered\_jobs.groupby(['Month', 'LOT\_V6\_OCCUPATION\_NAME'])['SALARY']  
 .mean().reset\_index()  
)  
  
fig = px.line(salary\_trend,   
 x="Month",   
 y="SALARY",   
 color="LOT\_V6\_OCCUPATION\_NAME",  
 title="Average Salary Trends Over Time for Top 5 Occupations")  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

/tmp/ipykernel\_2773/532501271.py:5: SettingWithCopyWarning:  
  
  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

Unable to display output for mime type(s): text/html

The line chart illustrates average salary trends over time for the top five most frequently posted occupations. A few meaningful patterns emerge:

* Computer Systems Engineer / Architect consistently ranks as the highest-paid occupation, maintaining an average salary around or above $150,000. This reflects the strong demand for highly skilled professionals in systems architecture, a field that supports infrastructure in both legacy enterprises and cloud-native environments.
* Data / Data Mining Analysts and Business Intelligence Analysts both show stable and competitive salaries in the range of ~$120,000–$130,000. These roles are closely tied to data-driven decision-making, reflecting how AI and analytics continue to shape business strategy and operations.
* Clinical Analysts / Clinical Documentation Specialists demonstrate slightly lower salary levels but remain relatively consistent, indicating steady demand in the healthcare and life sciences sectors—often associated with electronic health records, compliance, and process optimization.
* Business / Management Analysts show moderate but stable pay, aligning with generalist consulting and strategic support functions. While their salaries are slightly below the technical roles, they still remain above the $100,000 mark.

Overall, this plot reinforces the idea that technical and analytical occupations—especially those connected to data, engineering, and system-level design—continue to command premium salaries in the job market. Notably, salary stability across all five roles suggests that these are high-value, high-demand positions, resilient to short-term economic shifts.

## Average Salary by Employment Type

avg\_salary\_by\_type = (  
 job\_postings.groupby("EMPLOYMENT\_TYPE\_NAME")["SALARY"]  
 .mean()  
 .sort\_values(ascending=False)  
 .reset\_index()  
)  
  
import plotly.express as px  
  
fig = px.bar(avg\_salary\_by\_type,   
 x="EMPLOYMENT\_TYPE\_NAME",   
 y="SALARY",  
 title="Average Salary by Employment Type",  
 labels={"SALARY": "Average Salary ($)", "EMPLOYMENT\_TYPE\_NAME": "Employment Type"},  
 text="SALARY")  
  
fig.update\_layout(yaxis\_tickprefix="$", height=500)  
fig.update\_layout(\*\*plotly\_layout)  
fig.show()

Unable to display output for mime type(s): text/html

This bar chart compares the average salaries across different employment types, revealing key patterns in compensation based on job structure:

* Full-time roles (>32 hours) lead with the highest average salary at approximately $117,324, which aligns with expectations — these positions often come with more responsibilities, benefits, and long-term career opportunities.
* Part-time / full-time hybrid roles earn slightly less on average (~$104,379), potentially due to inconsistent hours or project-based employment models that offer flexibility but not always the highest compensation.
* Part-time roles (≤32 hours) average just below $102,000, a surprisingly competitive figure. This could reflect specialized part-time positions (e.g., consultants or contract professionals) that still command high hourly rates despite reduced hours.

Notably, the relatively narrow gap between employment types suggests that skills and job function may have a stronger influence on salary than hours alone. High-paying part-time and hybrid roles could indicate a shift toward flexible, high-skill labor markets, where experienced professionals negotiate premium pay for reduced workloads.

# Skill Gap Analysis

title: Skill Gap Analysis subtitle: Compare the skills required in IT job postings against the actual skills of your group members to identify knowledge gaps and areas for improvement. author: - name: Furong Wang affiliations: - id: bu name: Boston University city: Boston state: MA - name: Marco Perez Garcia affiliations: - ref: bu bibliography: references.bib csl: csl/econometrica.csl format: html: toc: true number-sections: true df-print: paged jupyter: python3

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import missingno as msno  
from collections import Counter  
import json  
import ast

# **Team-based Skill Dataframe**

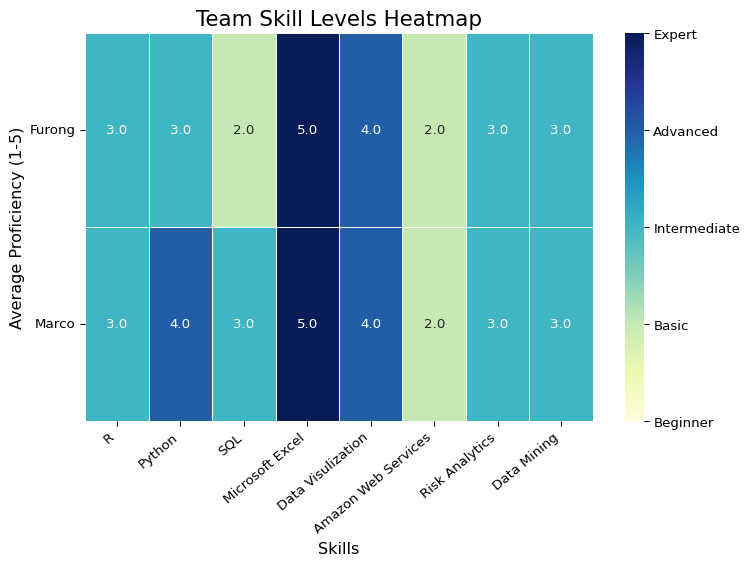
With our chosen IT career path as Business Analysts, we identified our current skills relevant to the role and assessed proficiency levels using a numerical scale from 1 to 5:  
1 = Beginner  
2 = Basic Knowledge  
3 = Intermediate  
4 = Advanced  
5 = Expert

The following heatmap visualized our team strengths and gaps.

job\_postings = pd.read\_csv("lightcast\_job\_postings.csv", low\_memory = False)

team\_skills\_data = {  
 "Name": ["Furong", "Marco"],  
 "R": [3, 3],  
 "Python": [3, 4],  
 "SQL": [2, 3],  
 "Microsoft Excel": [5, 5],  
 "Data Visulization": [4, 4],  
 "Amazon Web Services": [2, 2],  
 "Risk Analytics": [3, 3],  
 "Data Mining": [3, 3]  
}  
  
df\_team\_skills = pd.DataFrame(team\_skills\_data)  
df\_team\_skills.set\_index("Name", inplace = True)

plt.figure(figsize = (8, 6))  
heatmap = sns.heatmap(df\_team\_skills, annot = True, cmap = "YlGnBu",   
 linewidths = 0.5, fmt = ".1f", vmin = 1, vmax = 5)  
  
cbar = heatmap.collections[0].colorbar  
cbar.set\_ticks([1, 2, 3, 4, 5])  
cbar.set\_ticklabels(['Beginner', 'Basic', 'Intermediate', 'Advanced', 'Expert'])  
  
plt.title("Team Skill Levels Heatmap", fontsize = 16)  
plt.ylabel("Average Proficiency (1-5)", fontsize = 12)  
plt.xlabel("Skills", fontsize = 12)  
plt.xticks(rotation = 40, ha = 'right')  
plt.yticks(rotation = 0, ha = 'right')  
plt.tight\_layout()  
plt.show()



# **Team Skills vs. Industry Requirements**

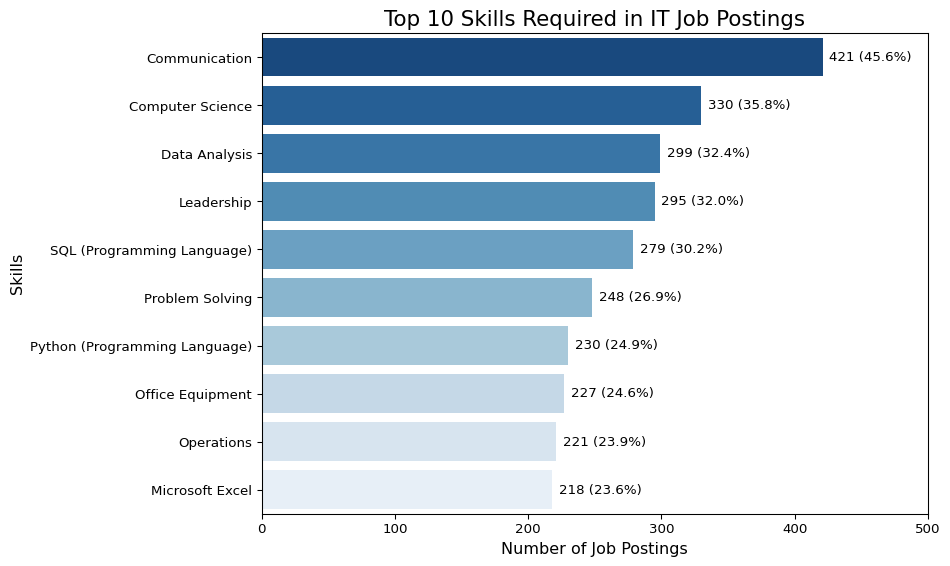
To compare our team’s skills to industry requirements, we identified the most in-demand skills from IT job postings. We focused on the industry group Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services, as it closely aligns with our chosen career path.

The bar plot below illustrates the top 10 skills most in demand within IT job postings, providing insights into industry expectations.

it\_jobs = job\_postings[job\_postings['NAICS\_2022\_6'] == 518210]

all\_skills = []  
  
def parse\_skills(skills\_str):  
 try:  
 if pd.isna(skills\_str):  
 return []  
 try:  
 return json.loads(skills\_str)  
 except:  
 return ast.literal\_eval(skills\_str)  
 except:  
 print(f"Warning: Could not parse skills: {skills\_str}")  
 return []  
  
for skills\_str in it\_jobs['SKILLS\_NAME'].dropna():  
 skills\_list = parse\_skills(skills\_str)  
 all\_skills.extend(skills\_list)  
  
skill\_counter = Counter(all\_skills)  
top\_10\_skills = skill\_counter.most\_common(10)  
  
top\_skills\_df = pd.DataFrame(top\_10\_skills, columns = ['Skill', 'Count'])

total\_postings = len(it\_jobs)  
top\_skills\_df['Percentage'] = (top\_skills\_df['Count'] / total\_postings \* 100).round(1)  
top\_skills\_df = top\_skills\_df.sort\_values('Count', ascending = False)  
  
plt.figure(figsize=(10, 6))  
sns.barplot(  
 x = 'Count',   
 y = 'Skill',   
 data = top\_skills\_df,  
 hue = 'Skill',  
 palette = "Blues\_r"  
)  
plt.title("Top 10 Skills Required in IT Job Postings", fontsize = 16)  
plt.xlabel("Number of Job Postings", fontsize = 12)  
plt.ylabel("Skills", fontsize = 12)  
plt.xlim(0, 500)   
  
for i, row in enumerate(top\_skills\_df.itertuples()):  
 plt.text(  
 row.Count + 5,   
 i,   
 f"{row.Count} ({row.Percentage}%)",  
 va='center'  
 )  
  
plt.tight\_layout()  
plt.show()



# **Team Skills Improvement Plan**

The bar plot of the top 10 most in-demand skills from IT job postings underscores the significance of core technical skills such as **Computer Science (35.8%)**, **Data Analysis (32.4%)**, **SQL (30.2%)**, and **Python (24.9%)**. These skills highlight the industry’s emphasis on data-driven decision-making and coding expertise as fundamental requirements.

When compared to our team’s current skill set, it is evident that we should prioritize developing knowledge in Data Science and SQL, which are highly demanded by the industry. Furthermore, while our team assessment did not initially account for **Communication**, the plot shows it as the most sought-after skill in IT job postings. This serves as a crucial reminder of the importance of teamwork, collaboration, and effective interpersonal interactions in IT careers. Technical expertise alone is insufficient; the ability to articulate ideas and communicate solutions is critical.

**The data also reveals that employers value a combination of technical proficiency, soft skills, and operational knowledge. To address skill gaps within our team, we can adopt a collaborative approach:**

1. Knowledge-sharing and mentorship: Team members skilled in communication can guide others in developing interpersonal abilities, while those with expertise in Python or SQL can lead technical workshops.
2. Specialization by strengths: Individuals strong in communication can take on tasks such as presentations and stakeholder engagement, whereas those skilled in technology can focus on back-end technical work.

This strategy not only bridges skill gaps but also optimizes team collaboration, positioning us as more competitive in the IT job market.

# Machine Learning Methods

title: Machine Learning Methods subtitle: Clustering and Machine Learning Techniques for Job Market Trends Analysis author: - name: Furong Wang affiliations: - id: bu name: Boston University city: Boston state: MA - name: Marco Perez Garcia affiliations: - ref: bu bibliography: references.bib csl: csl/econometrica.csl format: html: toc: true number-sections: true df-print: paged jupyter: python3

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import plotly.express as px  
from pyspark.sql import SparkSession  
import plotly.io as pio  
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler, StandardScaler  
from pyspark.ml import Pipeline  
from tabulate import tabulate  
from IPython.display import HTML  
from pyspark.sql import Window  
from pyspark.sql import functions as F  
from pyspark.sql.functions import col, pow, sqrt, abs, mean, avg, sum as spark\_sum, round as spark\_round, row\_number  
from pyspark.sql.types import DoubleType  
from pyspark.ml.regression import RandomForestRegressor  
from pyspark.ml.evaluation import RegressionEvaluator  
from pyspark.ml.clustering import KMeans  
from pyspark.ml.evaluation import ClusteringEvaluator  
import plotly.graph\_objects as go

plotly\_layout = dict(  
 font=dict(family="Arial", size=14),  
 title\_font=dict(size=20, family="Arial", color="black"),  
 paper\_bgcolor="white",  
 plot\_bgcolor="white",  
 margin=dict(t=60, l=60, r=30, b=60),  
 legend=dict(bordercolor="lightgray", borderwidth=1),  
 xaxis=dict(  
 title\_font=dict(size=16, color="black"),   
 tickfont=dict(size=12, color="black"),   
 showgrid=True,   
 gridcolor="lightgray",  
 showline=True,  
 linecolor="black",  
 linewidth=1,  
 mirror=True,  
 zeroline=True,  
 zerolinecolor="gray",  
 zerolinewidth=1  
 ),  
 yaxis=dict(  
 title\_font=dict(size=16, color="black"),   
 tickfont=dict(size=12, color="black"),   
 showgrid=True,   
 gridcolor="lightgray",  
 showline=True,  
 linecolor="black",  
 linewidth=1,  
 mirror=True,  
 zeroline=True,  
 zerolinecolor="gray",  
 zerolinewidth=1  
 ),  
)

# **Unsupervised Learning: KMeans Clustering**

In this section, we used KMeans clustering to group job postings based on Minimum Years of Experience and Salary. Our goal was to find natural patterns in how job roles are distributed across different experience and salary ranges. After clustering, we used NAICS6\_NAME industry classifications to interpret the types of industries represented in each cluster.

By doing so, we can better understand how different industries vary in their experience requirements and compensation levels, providing insights into salary structures across the job market.

spark = SparkSession.builder.appName("LightcastData").getOrCreate()  
  
# Load Data  
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","true").option("escape", "\"").csv("lightcast\_job\_postings.csv")  
  
# Show Schema and Sample Data  
# df.printSchema()   
# df.show(5)  
  
# Register the DataFrame as a temporary SQL table  
df.createOrReplaceTempView("job\_postings")

Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
25/04/30 02:58:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
[Stage 0:> (0 + 1) / 1] [Stage 1:> (0 + 1) / 1] 25/04/30 02:58:23 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

df = df.dropna(subset=['MIN\_YEARS\_EXPERIENCE', 'SALARY', 'NAICS6\_NAME'])  
df = df.filter(df.NAICS6\_NAME != 'Unclassified Industry')  
  
df\_casted = df.select(  
 col("MIN\_YEARS\_EXPERIENCE").cast(DoubleType()),  
 col("SALARY").cast(DoubleType()),  
 col("NAICS6\_NAME")  
)  
  
assembler = VectorAssembler(  
 inputCols=["MIN\_YEARS\_EXPERIENCE", "SALARY"],   
 outputCol='features\_unscaled'  
)  
df\_features = assembler.transform(df\_casted)

scaler = StandardScaler(  
 inputCol='features\_unscaled',   
 outputCol='features',   
 withMean=True,   
 withStd=True  
)  
scaler\_model = scaler.fit(df\_features)  
df\_scaled = scaler\_model.transform(df\_features)

[Stage 2:> (0 + 1) / 1]

scores = []  
ks = list(range(2, 9))  
  
for k in ks:  
 kmeans = KMeans().setK(k).setSeed(42).setFeaturesCol("features")  
 model = kmeans.fit(df\_scaled)  
 transformed = model.transform(df\_scaled)  
  
 evaluator = ClusteringEvaluator(  
 featuresCol="features", predictionCol="prediction", metricName="silhouette")  
 score = evaluator.evaluate(transformed)  
 scores.append(score)  
  
fig = go.Figure()  
fig.add\_trace(go.Scatter(x=ks, y=scores, mode='lines+markers', name='Silhouette Score'))  
fig.update\_layout(title="Silhouette Score vs. k",   
 xaxis\_title="k",  
 yaxis\_title="Score",  
 template="simple\_white",  
 width=900)  
fig.update\_layout(\*\*plotly\_layout)   
fig.show()

[Stage 5:> (0 + 1) / 1] [Stage 8:> (0 + 1) / 1] [Stage 42:> (0 + 1) / 1] [Stage 45:> (0 + 1) / 1] [Stage 47:> (0 + 1) / 1] [Stage 50:> (0 + 1) / 1] [Stage 53:> (0 + 1) / 1] [Stage 101:> (0 + 1) / 1] [Stage 104:> (0 + 1) / 1] [Stage 106:> (0 + 1) / 1] [Stage 109:> (0 + 1) / 1] [Stage 112:> (0 + 1) / 1] [Stage 160:> (0 + 1) / 1] [Stage 163:> (0 + 1) / 1] [Stage 165:> (0 + 1) / 1] [Stage 168:> (0 + 1) / 1] [Stage 171:> (0 + 1) / 1] [Stage 219:> (0 + 1) / 1] [Stage 222:> (0 + 1) / 1] [Stage 224:> (0 + 1) / 1] [Stage 227:> (0 + 1) / 1] [Stage 230:> (0 + 1) / 1] [Stage 278:> (0 + 1) / 1] [Stage 281:> (0 + 1) / 1] [Stage 283:> (0 + 1) / 1] [Stage 286:> (0 + 1) / 1] [Stage 289:> (0 + 1) / 1] [Stage 325:> (0 + 1) / 1] [Stage 328:> (0 + 1) / 1] [Stage 330:> (0 + 1) / 1] [Stage 333:> (0 + 1) / 1] [Stage 336:> (0 + 1) / 1] [Stage 384:> (0 + 1) / 1] [Stage 387:> (0 + 1) / 1] [Stage 389:> (0 + 1) / 1]

Unable to display output for mime type(s): text/html

kmeans\_final = KMeans(featuresCol='features', k=4, seed=688)  
model\_final = kmeans\_final.fit(df\_scaled)  
  
predictions = model\_final.transform(df\_scaled)

[Stage 392:> (0 + 1) / 1] [Stage 395:> (0 + 1) / 1] [Stage 433:> (0 + 1) / 1]

industry\_counts = predictions.groupBy('prediction', 'NAICS6\_NAME').count()  
  
window\_spec = Window.partitionBy('prediction')  
industry\_counts = industry\_counts.withColumn('total', spark\_sum('count').over(window\_spec))  
  
industry\_counts = industry\_counts.withColumn('percentage', spark\_round(col('count') / col('total') \* 100, 2))  
  
window\_top5 = Window.partitionBy('prediction').orderBy(col('percentage').desc())  
industry\_top5 = industry\_counts.withColumn('row\_num', row\_number().over(window\_top5)).filter(col('row\_num') <= 5)  
  
industry\_top5.orderBy('prediction', 'row\_num').show(100, truncate=False)

[Stage 436:> (0 + 1) / 1]

+----------+--------------------------------------------------------------------+-----+-----+----------+-------+  
|prediction|NAICS6\_NAME |count|total|percentage|row\_num|  
+----------+--------------------------------------------------------------------+-----+-----+----------+-------+  
|0 |Computer Systems Design Services |1267 |4167 |30.41 |1 |  
|0 |Administrative Management and General Management Consulting Services|276 |4167 |6.62 |2 |  
|0 |Custom Computer Programming Services |188 |4167 |4.51 |3 |  
|0 |Direct Health and Medical Insurance Carriers |167 |4167 |4.01 |4 |  
|0 |Drugs and Druggists' Sundries Merchant Wholesalers |165 |4167 |3.96 |5 |  
|1 |Administrative Management and General Management Consulting Services|698 |2598 |26.87 |1 |  
|1 |Web Search Portals and All Other Information Services |234 |2598 |9.01 |2 |  
|1 |Employment Placement Agencies |146 |2598 |5.62 |3 |  
|1 |Offices of Certified Public Accountants |141 |2598 |5.43 |4 |  
|1 |Commercial Banking |128 |2598 |4.93 |5 |  
|2 |Administrative Management and General Management Consulting Services|480 |7102 |6.76 |1 |  
|2 |Employment Placement Agencies |438 |7102 |6.17 |2 |  
|2 |Colleges, Universities, and Professional Schools |385 |7102 |5.42 |3 |  
|2 |Direct Health and Medical Insurance Carriers |370 |7102 |5.21 |4 |  
|2 |Custom Computer Programming Services |353 |7102 |4.97 |5 |  
|3 |Administrative Management and General Management Consulting Services|1303 |7390 |17.63 |1 |  
|3 |Employment Placement Agencies |465 |7390 |6.29 |2 |  
|3 |Commercial Banking |408 |7390 |5.52 |3 |  
|3 |Direct Health and Medical Insurance Carriers |364 |7390 |4.93 |4 |  
|3 |Computer Systems Design Services |358 |7390 |4.84 |5 |  
+----------+--------------------------------------------------------------------+-----+-----+----------+-------+

pandas\_df = predictions.select('MIN\_YEARS\_EXPERIENCE', 'SALARY', 'prediction').toPandas()  
  
pandas\_df['Cluster\_Name'] = pandas\_df['prediction'].map({0: 'Cluster 1',   
 1: 'Cluster 2',   
 2: 'Cluster 3',  
 3: 'Cluster 4'})  
  
plt.figure(figsize=(8, 5))  
fig = px.scatter(  
 pandas\_df,   
 x="MIN\_YEARS\_EXPERIENCE",   
 y="SALARY",   
 color="Cluster\_Name",   
 title="K-Means Clustering on Job Postings Data",   
 labels={  
 "MIN\_YEARS\_EXPERIENCE": "Minimum Years of Experience",   
 "SALARY": "Salary",  
 "Cluster\_Name": "Cluster"  
 },  
 category\_orders={"Cluster\_Name": ["Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4"]}  
)  
  
fig.update\_layout(\*\*plotly\_layout,  
 width=800,  
 height=500)   
fig.show()

[Stage 442:> (0 + 1) / 1]

Unable to display output for mime type(s): text/html

<Figure size 768x480 with 0 Axes>

Based on the silhouette score of the K value and practical anlysis requirements, we identified four distinct clusters that capture major compensation patterns in the labor market.

**Here are key findings based on salary and experience trends:**

* **Cluster 1:**
  + **Experience/Salary Pattern:** Requires higher minimum years of experience. Offers only moderate salary levels despite higher experience.
  + **Top Industries:** Computer Systems Design Services, Administrative Management and General Management Consulting Services, and Custom Computer Programming Services.
  + **Insight:** Jobs demanding significant prior experience but offering relatively moderate compensation. Indicates competitive markets in tech and consulting sectors.
* **Cluster 2:**
  + **Experience/Salary Pattern:** Consistently the highest salaries across a wide range of experience levels.
  + **Top Industries:** Administrative Management and General Management Consulting Services, Web Search Portals and Other Information Services, and Commercial Banking.
  + **Insight:** Reflects premium-paying roles in consulting, web services, and finance. This suggests opportunities for substantial earnings even with moderate experience.
* **Cluster 3:**
  + **Experience/Salary Pattern:** Requires lower years of experience. Salary levels are generally the lowest.
  + **Top Industries:** Administrative Management and General Management Consulting Services, Employment Placement Agencies, and Direct Health and Medical Insurance Carriers.
  + **Insight:** Entry-level or early-career roles in sectors with limited immediate salary growth.
* **Cluster 4:**
  + **Experience/Salary Pattern:** Moderate years of experience required. Salary levels are moderately high.
  + **Top Industries:** Administrative Management and General Management Consulting Services, Employment Placement Agencies, and Commercial Banking.
  + **Insight:** Steady career tracks offering good compensation for mid-experience professionals.

**Implications for Salary and Compensation Trends:**  
- Salary growth is not always linear with experience; certain clusters show salary plateaus despite increasing experience.  
- Industry effects are significant: sectors like Professional Services and Finance consistently appear across clusters, but compensation levels vary depending on experience requirements.  
- High-paying opportunities exist both at low and high experience levels, depending on industry and role specialization.

**Implications for Job Seekers:**  
- **High Salary Aspirations:** Target roles in Cluster 2 industries like consulting, finance, and web services where premium salaries are achievable even with moderate experience.

* **Career Launch:** Cluster 3 industries may provide easier entry points for new graduates but with lower starting salaries. In contrast, positions in Cluster 4 offer a good balance between experience investment and salary rewards.
* **Beware of High-Experience/Moderate-Pay Sectors:** Cluster 1 jobs may require significant experience without corresponding salary premiums, requiring careful career planning.

# **Supervised Learning: Random Forest Regression**

To deepen our analysis on Salary and Compensation Trends, we constructed a Random Forest Regression model using salary as the target variable. The goal of this model is to predict salary outcomes based on key job posting attributes, and to identify the relative importance of different factors influencing compensation in the labor market.

The predictor variables selected for the model include: DURATION, MIN\_YEARS\_EXPERIENCE, LOT\_V6\_OCCUPATION\_NAME, STATE\_NAME, EMPLOYMENT\_TYPE\_NAME

df\_rf = df.dropna(subset=['DURATION', 'MIN\_YEARS\_EXPERIENCE',   
 'LOT\_V6\_OCCUPATION\_NAME', 'STATE\_NAME', 'EMPLOYMENT\_TYPE\_NAME',  
 'SALARY'])  
  
categorical\_cols = ['LOT\_V6\_OCCUPATION\_NAME', 'STATE\_NAME', 'EMPLOYMENT\_TYPE\_NAME']   
continuous\_cols = ['DURATION', 'MIN\_YEARS\_EXPERIENCE']   
  
# Index and One-Hot Encode  
indexers = [StringIndexer(inputCol=col, outputCol=f"{col}\_idx", handleInvalid='skip') for col in categorical\_cols]  
encoders = [OneHotEncoder(inputCol=f"{col}\_idx", outputCol=f"{col}\_vec") for col in categorical\_cols]  
  
# Assemble base features   
assembler = VectorAssembler(  
 inputCols=continuous\_cols   
 + [f"{col}\_vec" for col in categorical\_cols],   
 outputCol="features"  
)  
  
# Build pipeline and transform  
pipeline = Pipeline(stages=indexers + encoders + [assembler])   
data = pipeline.fit(df\_rf).transform(df\_rf)  
  
# Show final structure  
data.select("SALARY", "features").show(5, truncate=False)

[Stage 443:> (0 + 1) / 1] [Stage 446:> (0 + 1) / 1] [Stage 449:> (0 + 1) / 1]

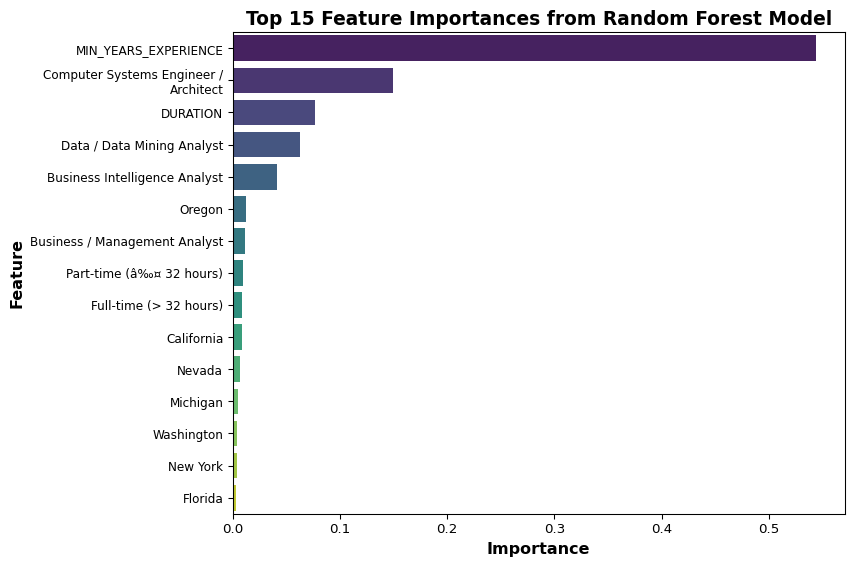
+------+------------------------------------------+  
|SALARY|features |  
+------+------------------------------------------+  
|192800|(59,[0,1,2,15,57],[55.0,6.0,1.0,1.0,1.0]) |  
|125900|(59,[0,1,2,16,57],[18.0,12.0,1.0,1.0,1.0])|  
|118560|(59,[0,1,3,28,57],[20.0,5.0,1.0,1.0,1.0]) |  
|192800|(59,[0,1,2,13,57],[55.0,6.0,1.0,1.0,1.0]) |  
|116500|(59,[0,1,2,16,57],[16.0,12.0,1.0,1.0,1.0])|  
+------+------------------------------------------+  
only showing top 5 rows

train\_data, test\_data = data.randomSplit([0.8, 0.2], seed=688)  
  
rf = RandomForestRegressor(featuresCol="features",  
 labelCol="SALARY",   
 numTrees=150,  
 maxDepth=9,   
 seed=688   
 )  
  
# Train model  
rf\_model = rf.fit(train\_data.select("SALARY", "features"))  
  
# Generate predictions  
rf\_preds = rf\_model.transform(train\_data.select("SALARY", "features"))

[Stage 453:> (0 + 1) / 1] [Stage 454:> (0 + 1) / 1] [Stage 455:> (0 + 1) / 1] [Stage 457:> (0 + 1) / 1] [Stage 461:> (0 + 1) / 1] [Stage 463:> (0 + 1) / 1] [Stage 465:> (0 + 1) / 1] 25/04/30 03:05:02 WARN DAGScheduler: Broadcasting large task binary with size 1634.7 KiB  
[Stage 467:> (0 + 1) / 1] 25/04/30 03:05:05 WARN DAGScheduler: Broadcasting large task binary with size 2.6 MiB  
[Stage 469:> (0 + 1) / 1][Stage 470:> (0 + 1) / 1] 25/04/30 03:05:08 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB  
[Stage 471:> (0 + 1) / 1][Stage 472:> (0 + 1) / 1] 25/04/30 03:05:12 WARN DAGScheduler: Broadcasting large task binary with size 5.7 MiB  
[Stage 473:> (0 + 1) / 1][Stage 473:=========================================================(1 + 0) / 1][Stage 474:> (0 + 1) / 1] WARNING: An illegal reflective access operation has occurred  
WARNING: Illegal reflective access by org.apache.spark.util.SizeEstimator$ (file:/opt/spark-3.5.5-bin-hadoop3/jars/spark-core\_2.12-3.5.5.jar) to field java.nio.charset.Charset.name  
WARNING: Please consider reporting this to the maintainers of org.apache.spark.util.SizeEstimator$  
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations  
WARNING: All illegal access operations will be denied in a future release

# Extract feature importances  
def get\_actual\_feature\_names(df\_rf, assembler, encoded\_cols):  
 full\_feature\_names = []  
  
 for col\_name in assembler.getInputCols():  
 if col\_name in encoded\_cols:  
 try:  
 attr\_meta = df\_rf.schema[col\_name].metadata['ml\_attr']['attrs']  
 for attr\_group in attr\_meta.values():  
 for attr in attr\_group:  
 full\_feature\_names.append(attr['name'])  
 except:  
 full\_feature\_names.append(col\_name)  
 else:  
 full\_feature\_names.append(col\_name)  
  
 return full\_feature\_names  
  
encoded\_cols = [f"{col}\_vec" for col in categorical\_cols]   
feature\_names = get\_actual\_feature\_names(data, assembler, encoded\_cols)  
importances = rf\_model.featureImportances.toArray()

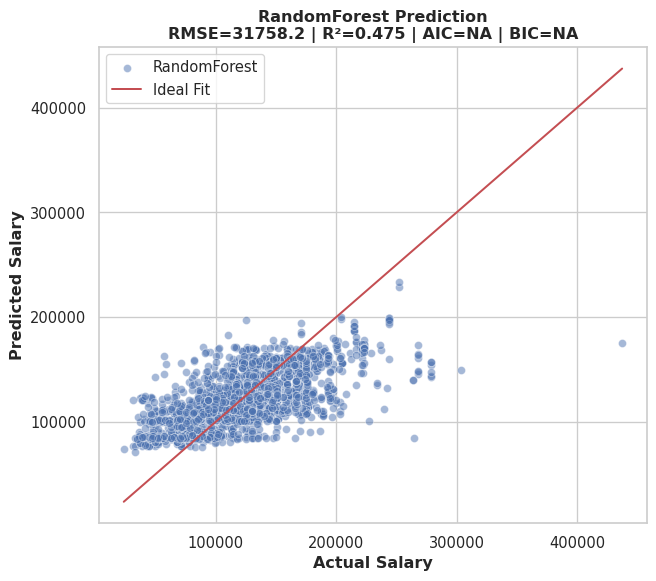
def clean\_feature\_names(feature\_list):  
 clean\_names = []  
 for name in feature\_list:  
 if isinstance(name, list):  
 clean\_names.append(", ".join(str(n) for n in name))  
 elif isinstance(name, str) and name.startswith("["):  
 clean\_names.append(name.replace("[", "").replace("]", "").replace("'", "").replace('"', '').strip())  
 else:  
 clean\_names.append(str(name))  
 return clean\_names  
  
# Build dataframe  
importance\_df = pd.DataFrame({  
 "Feature": feature\_names,  
 "Importance": importances  
}).sort\_values(by="Importance", ascending=False)  
  
importance\_df["Feature"] = clean\_feature\_names(importance\_df["Feature"])  
top\_importance\_df = importance\_df.head(15)  
  
# Plot  
plt.figure(figsize=(9, 6))  
sns.barplot(  
 x="Importance",  
 y="Feature",  
 data=top\_importance\_df,  
 hue="Feature",  
 palette="viridis"  
)  
  
import textwrap  
labels = plt.gca().get\_yticklabels()  
new\_labels = [textwrap.fill(label.get\_text(), width=30) for label in labels]  
plt.yticks(range(len(new\_labels)), new\_labels, fontsize=9)  
plt.xticks(fontsize=10)  
  
plt.title("Top 15 Feature Importances from Random Forest Model", fontsize=14, fontweight='bold')  
plt.xlabel("Importance", fontsize=12, fontweight='bold')  
plt.ylabel("Feature", fontsize=12, fontweight='bold')  
plt.tight\_layout()  
plt.show()



evaluator\_r2 = RegressionEvaluator(labelCol="SALARY", predictionCol="prediction", metricName="r2")  
  
rf\_preds = rf\_model.transform(test\_data)  
rf\_residuals = rf\_preds.select(  
 col("SALARY"),  
 col("prediction"),  
 (col("SALARY") - col("prediction")).alias("residual")  
)  
  
rf\_r2 = evaluator\_r2.evaluate(rf\_preds)   
rf\_rmse = np.sqrt(rf\_residuals.select(avg(pow(col("residual"), 2))).first()[0])   
rf\_aic = None  
rf\_bic = None  
  
rf\_pdf = rf\_residuals.select("SALARY", col("prediction").alias("RandomForest")).toPandas()  
rf\_df = pd.DataFrame({"SALARY": rf\_pdf["SALARY"], "RandomForest": rf\_pdf["RandomForest"]})

[Stage 475:> (0 + 1) / 1] [Stage 476:> (0 + 1) / 1] [Stage 479:> (0 + 1) / 1]

plt.figure(figsize=(7, 17))  
sns.set(style="whitegrid")  
  
models = {"RandomForest": (rf\_rmse, rf\_r2, "NA", "NA")}  
  
model\_dfs = {"RandomForest": rf\_df}  
  
for idx, (model\_name, (rmse, r2, aic, bic)) in enumerate(models.items(), 1):  
 plt.subplot(3, 1, idx)  
   
 model\_data = model\_dfs[model\_name]  
   
 sns.scatterplot(x="SALARY", y=model\_name, data=model\_data, alpha=0.5, label=model\_name)  
   
 x\_min = model\_data["SALARY"].min()  
 x\_max = model\_data["SALARY"].max()  
 plt.plot([x\_min, x\_max], [x\_min, x\_max], 'r-', label="Ideal Fit")  
   
 if aic != "NA" and bic != "NA":  
 plt.title(f"{model\_name} Prediction\nRMSE={rmse:.1f} | R²={r2:.3f} | AIC={aic:.1f} | BIC={bic:.1f}", fontweight="bold")  
 else:  
 plt.title(f"{model\_name} Prediction\nRMSE={rmse:.1f} | R²={r2:.3f} | AIC=NA | BIC=NA", fontweight="bold")  
   
 plt.xlabel("Actual Salary", fontweight="bold")  
 plt.ylabel("Predicted Salary", fontweight="bold")  
 plt.legend()  
  
plt.tight\_layout()  
plt.show()



**Model Evaluation:**  
- The R-Squared of 0.454 indicates a moderate level of predictive power, suggesting that the model captures a substantial portion of salary variability but leaves room for improvement.  
- The scatterplot shows that most predictions are reasonably aligned with the actual salaries but tend to underpredict higher salary values (especially above $200,000), which is common due to the small number of very high salaries (“long tail” effect).

**Feature Importance Analysis:**  
- The top 15 feature importances from random forest model show how different factors contribute to salary predictions. In our model, years of experience are by far the strongest predictor of salary. Having some certain occupation titles can also greatly affect salary expectations. Interestingly, the job posting duration is also an important factor in salary prediction, which may be related to stable roles.  
- On the other hand, geographic location and job type have less impact on salary predictions, with states like Oregon and California showing specific salary patterns.

**Implications for Job Seekers:**  
- **Experience Pays Off:** The model shows that minimum years of experience is the dominant factor influencing salary. For job seekers, gaining and accurately showcasing professional experience is crucial to achieving higher salary outcomes. Moreover, investing in data-related skills can be a smart career move.  
- **Occupation Choice Matters:** Specific technical roles (especially Computer Systems Engineers, Data Analysts, and Business Intelligence Analysts) are associated with higher salaries. Choosing high-demand, specialized roles can significantly improve salary prospects.  
- **Location Strategy:** While experience and occupation dominate, geography still plays a role, for example, states like Oregon, California, and New York affect salary expectations. Job seekers willing to relocate or negotiate for remote work with companies based in higher-paying states may gain salary advantages.

Chen, J., Z. Song, and C. H. Lam. (2024): “Unraveling the trajectory of data science salaries in the united states: A comprehensive analysis from 2020 to 2023 with future salary projections,” *Advances in Economics, Management and Political Sciences*, 92, 10–20.

Quan, T. Z., and M. Raheem. (2023): “Human resource analytics on data science employment based on specialized skill sets with salary prediction,” *International Journal of Data Science*, 4, 40–59.

Zhu, J. (2024): “Unveiling Salary Trends: Exploring Machine Learning Models for Predicting Data Science Job Salaries,” *2024 2nd international conference on image, algorithms and artificial intelligence (ICIAAI 2024)*, Atlantis Press, 173–82.